```
// "use strict";
// added in ES 5
// use this for vanilla Javascript

"use strict";

// 2. Variable(변수), re(read/write) -> 메모리에 값을 읽고 쓰기 가능
// let (added in ES 6)

// 각 변수는 메모리에 할당 됨
// 전역변수 사용시 어디서나 사용가능(항상 메모리에 탑재되어 있음)
// 클래스나 함수등 필요한 부분에서만 사용할 것
let globalName = "global name";
{
  /* name이라고 불리는(포인터가 가리키는) 박스(메모리)안에 macaron 이라는 내용을 저장 */
  let name = "macaron";
  console.log(name);
  name = "hello";
  console.log(name);
  console.log(globalName);
} /* Block Scope */
console.log(name); /* 지역변수에 접근해도 값이 나오지 않음 */
console.log(globalName); /* 전역변수에 접근해서 값이 나옴 */

// ES 6 이전 변수 선언 -> var(단 이제는 사용 지양할 것)
// JS는 값을 설정하지 않아도 값을 호출 할 수는 있음(var 이용) -> undefined 출력
/* var hoisting (move declaration from bottom to top) -> 때문에 발생
  호이스팅 : 어디에 선언했더라도 선언을 맨 위로 끌어올리는 것 */
// var 는 block scope도 없음 (block을 무시)
{
console.log(age);
age = 4;
var age;
}
console.log(age);

// Mutable type
// let 은 바로 에러메세지를 출력함
// ES 6 에서 추가 된 기능들은 can i use 에서 호환성 확인해 볼 것
/* age2 = 4;
let age2; */

// 3. Constant, r(read only) -> 읽기만 가능(자물쇠가 생김!)
// use const whenever possible -> JS에서는 특별한 경우 아니면 const 사용
// Immutable type
// favor immutable data type always for a few reasons:
//  - security
//  - thread safety
//  -> 프로세스 내부의 다양한 스레드들이 변수에 동시다발적으로 접근(값의 변경)가능
//  - reduce human mistakes
const daysInWeek = 7;
const maxNumber = 5;

// Variable types
/* primitive, single item(더 이상 작은 단위로 나눠질 수 없는 타입) -> 메모리에 값이 바로 저장
  -> number, string, boolean, null, undefiend, symbol*/
// object, box container(single type을 여러개 묶어 관리하는 타입) -> 오브젝트가 가리키는 곳에 레퍼런스가 저장되어 있음
// function, first-class function -> 변수 할당 가능, 함수의 인자(파라미터)로 전달가능, 함수 리턴값으로 사용가능

```

```
// JS에서는 number type 을 직접적으로 지정 가능(int,double... 필요 없이 바로)
let a = 12;
let b = 1.2;

const count = 17; // integer
const size = 17.1; //decimal number
console.log(`value : ${count}, type : ${typeof count}`);
console.log(`value : ${size}, type : ${typeof size}`);

// number
const infinity = 1 / 0;
const negativeInfinity = -1 / 0;
const nAn = "not a number" / 2; /* NaN */
console.log(infinity);
console.log(negativeInfinity);
console.log(nAn);

// bigInt (fairly new, don't use it yet) -> 숫자 맨 뒤에 n 추가(이런게 있다 정도로 알면 됨)
const bigInt = 12345678901234567890123456789012345678901234567890n; // over -2*53 ~ 2*53
console.log(`value : ${bigInt}, type : ${typeof bigInt}`);

// string -> 한 글자든 여러 글자든 string 으로 처리 (char = string)
const char = 'c';
const macaron = "macaron";
const hello = "Hello " + macaron;
console.log(`value : ${hello}, type : ${typeof hello}`);
const helloBob = `hi ${macaron}!`; //template literals(=string) -> 변수의 값이 자동으로 붙여나옴
console.log(`value : ${helloBob}, type : ${typeof helloBob}`);
// backtick 미사용시 이렇게 하나씩 설정해야 함
console.log("value : " + helloBob +  "type : " + typeof helloBob);

// boolean
// false : 0, null, undefined, NaN, '', ""(비어있는 문자열)
// ture : any other value
const canRead = true;
const test = 3 < 1; //false
console.log(`value : ${canRead}, type : ${typeof canRead}`);
console.log(`value : ${test}, type : ${typeof test}`);

// null -> 내가 명확하게 빈 공간인 값이라고 설정한 것(empty)
let nothing = null;
console.log(`value : ${nothing}, type : ${typeof nothing}`);

// undefined -> 선언은 되었지만 아무것도 값이 지정되지 않은 것(있는지 없는지 모름)
let x;
console.log(`value : ${x}, type : ${typeof x}`);

// symbol, create unique identifiers for objects
// 다른 자료구조에서 고유한 식별자가 필요할때나 우선순위를 구별할때 사용
const symbol1 = Symbol("id");
const symbol2 = Symbol("id");
console.log(symbol1 === symbol2); /* 동일한 string을 설정하여도 false가 나옴(고유한 식별자 생성) */
// Symbol.for -> 동일하게 설정하면 동일한 값으로 판단(true)
const gsymbol1 = Symbol.for("id");
const gsymbol2 = Symbol.for("id");
console.log(gsymbol1 === gsymbol2);
// symbol.description -> 그냥 symbol을 출력하면 에러 발생!
console.log(`value : ${symbol1.description}, type : ${typeof symbol1.description}`);

```

```
// object, real-life object, data structure
const ellie = {name : "ellie", age : 20};
// ellie가 가리키는 메모리의 포인터는 잠겨있어 다른 오브젝트로 할당 불가
// 하지만, ellie에 속한 변수들은 수정가능
ellie.age = 21;


// 5. Dynamic typing : dynamically typed language
let text = "hello";
console.log(text.charAt(0)) // h -> string으로 이해했기 때문에 예상가능
console.log(`value : ${text}, type : ${typeof text}`); // string
text = 1;
console.log(`value : ${text}, type : ${typeof text}`); // number
text = '7' + 5;
console.log(`value : ${text}, type : ${typeof text}`); // string (75)
text = '8' / '2';
console.log(`value : ${text}, type : ${typeof text}`); // number (4)
/* console.log(text.charAt(0)) */ // 에러발생 -> 갑자기 number로 바뀌어 예상이 힘듬
```